# Azure Redis Cache FAQ

Learn the answers to common questions, patterns and best practices for Azure Redis Cache.

## What Redis Cache offering and size should I use? Each Azure Redis Cache offering provides different levels of **size**, **bandwidth**, **high availability** and **SLA** options.

The following are considerations for choosing a Cache offering.

* **Memory**: The Basic and Standard tiers offer 250 MB – 53 GB. The Premium tier offers up to 530 GB with more available [on request](mailto:wapteams@microsoft.com?subject=Redis%20Cache%20quota%20increase). For more information see [Azure Redis Cache Pricing](/home/features/cache/#price).
* **Network Performance**: If you have a workload that requires high throughput the Premium tier offers more bandwidth compared to Standard or Basic. Also within each tier larger sizes caches have more bandwidth because of the underlying VM that hosts the cache. Please see the following table for more information.
* **Throughput**: The Premium tier offers the maximum available throughput. If the cache server or client reaches the bandwidth limits, you will receive timeouts on the client side. Please see the following table for more information.
* **High Availability/SLA**: Azure Redis Cache guarantees that a Standard/Premium (no SLA for Premium until after the preview period) cache will be available at least 99.9% of the time. To learn more about our SLA, see [Azure Redis Cache Pricing](/home/features/cache/#price). The SLA only covers connectivity to the Cache endpoints. The SLA does not cover protection from data loss. We recommend using the Redis data persistence feature in the Premium tier to increase resiliency against data loss.
* **Redis Data Persistence**: The Premium tier allows you to persist the cache data in an Azure Storage account. In a Basic/Standard cache all the data is stored only in memory. In case of underlying infrastructure issues there can be potential data loss. We recommend using the Redis data persistence feature in the Premium tier to increase resiliency against data loss. Azure Redis Cache offers RDB and AOF (coming soon) options in Redis persistence. For more information, see [How to configure persistence for a Premium Azure Redis Cache](/documentation/articles/cache-how-to-premium-persistence).
* **Redis Cluster**: If you want to create caches larger than 53 GB or want to shard data across multiple Redis nodes, you can use Redis clustering which is available in the Premium tier. Each node consists of a primary/replica cache pair for high availability. For more information, see [How to configure clustering for a Premium Azure Redis Cache](/documentation/articles/cache-how-to-premium-clustering).
* **Enhanced security and network isolation**: Azure Virtual Network (VNET) deployment provides enhanced security and isolation for your Azure Redis Cache, as well as subnets, access control policies, and other features to further restrict access. For more information, see [How to configure Virtual Network support for a Premium Azure Redis Cache](/documentation/articles/cache-how-to-premium-vnet).
* **Configure Redis**: In both the Standard and Premium tiers, you can configure Redis for Keyspace notifications.
* **Maximum number of client connections**: The Premium tier offers the maximum number of clients that can connect to Redis, with a higher number of connections for larger sized caches. [Please refer to the pricing page for details](/home/features/cache/#price).
* **Dedicated Core for Redis Server**: In the Premium tier all cache sizes have a dedicated core for Redis. In the Basic/Standard tiers the C1 size and above have a dedicated core for Redis server.
* **Redis is single-threaded** so having more than two cores does not provide additional benefit over having just two cores, but larger VM sizes typically have more bandwidth than smaller sizes. If the cache server or client reaches the bandwidth limits, then you will receive timeouts on the client side.
* **Performance improvements**: Caches in the Premium tier are deployed on hardware which have faster processors and gives better performance compared to the Basic or Standard tier. Premium tier Caches have higher throughput and lower latencies.

The following table shows the maximum bandwidth values observed while testing various sizes of Standard and Premium caches using redis-benchmark.exe from an Iaas VM against the Azure Redis Cache endpoint. Note that these values are not guaranteed and there is no SLA for these numbers, but should be typical. You should load test your own application to determine the right cache size for your application.

From this table we can draw the following conclusions.

* Throughput for the same sized Cache is higher in Premium as compared to Standard tier. Eg. For a 6 GB Cache, throughput of P1 is 140K RPS as compared to 49K for C3.
* With Redis clustering, throughput increases linearly as you increase the number of shards (nodes) in the cluster. Eg. If you create a P4 cluster of 10 shards, then the available throughput is 250K \*10 = 2.5 Million RPS
* Throughput for bigger key sizes is higher in Premium tier as compared to Standard Tier.

|  |  |  |  |
| --- | --- | --- | --- |
| Pricing tier | Size | Available bandwidth (Mbps) | 1 KB Key size |
| **Standard cache sizes** |  |  | **Requests per second (RPS)** |
| C0 | 250 MB | 5 | 600 |
| C1 | 1 GB | 100 | 12200 |
| C2 | 2.5 GB | 200 | 24000 |
| C3 | 6 GB | 400 | 49000 |
| C4 | 13 GB | 500 | 61000 |
| C5 | 26 GB | 1000 | 115000 |
| C6 | 53 GB | 2000 | 150000 |
| **Premium cache sizes** |  |  | **Requests per second (RPS), per shard** |
| P1 | 6 GB | 1000 | 140000 |
| P2 | 13 GB | 2000 | 220000 |
| P3 | 26 GB | 2000 | 220000 |
| P4 | 53 GB | 4000 | 250000 |

For instructions on downloading the Redis tools such as redis-benchmark.exe, see the [How can I run Redis commands?](#cache-commands) section.

## In what region should I locate my cache?

For best performance and lowest latency, locate your Azure Redis Cache in the same region as your cache client application.

## How am I billed for Azure Redis Cache?

Azure Redis Cache pricing is [here](/home/features/cache/#price). The pricing page lists pricing as an hourly rate. Caches are billed on a per-minute basis from the time that the cache is created until the time that a cache is deleted. There is no option for stopping or pausing the billing of a cache.

## Why am I seeing timeouts?

Timeouts happen in the client that you use to talk to Redis. For the most part Redis server does not time out. When a command is sent to the Redis server, the command is queued up and Redis server eventually picks up the command and executes it. However the client can time out during this process and if it does an exception is raised on the calling side. For more information on troubleshooting timeout issues, see [Investigating timeout exceptions in StackExchange.Redis for Azure Redis Cache](http://azure.microsoft.com/blog/2015/02/10/investigating-timeout-exceptions-in-stackexchange-redis-for-azure-redis-cache/).

## How do I monitor the health and performance of my cache?

Windows Azure Redis Cache instances can be monitored in the [Azure preview portal](https://manage.windowsazure.cn). You can view metrics, pin metrics charts to the Startboard, customize the date and time range of monitoring charts, add and remove metrics from the charts, and set alerts when certain conditions are met. These tools enable you to monitor the health of your Azure Redis Cache instances and help you manage your caching applications. For more information on monitoring your caches, see [Monitor Azure Redis Cache](https://msdn.microsoft.com/zh-cn/library/azure/dn763945.aspx).

## Why was my client disconnected from the cache?

The following are some common reason for a cache disconnect.

* Client-side causes
  + The client application was redeployed.
  + The client application performed a scaling operation.
    - In the case of Cloud Services or Web Apps, this may be due to auto-scaling.
  + The networking layer on the client side changed.
  + Transient errors occurred in the client or in the network nodes between the client and the server.
  + The bandwidth threshold limits were reached.
  + CPU bound operations took too long to complete.
* Server-side causes
  + On the standard cache offering, the Azure Redis Cache service initiated a fail-over from the primary node to the secondary node.
  + Azure was patching the instance where the cache was deployed
    - This can be for Redis server updates or general VM maintenance.

## What do the StackExchange.Redis configuration options do?

StackExchange.Redis has many options. This section talks about some of the common settings. For more detailed information about StackExchange.Redis options, see [StackExchange.Redis configuration](https://github.com/StackExchange/StackExchange.Redis/blob/master/Docs/Configuration.md).

|  |  |  |
| --- | --- | --- |
| ConfigurationOptions | Description | Recommendation |
| AbortOnConnectFail | When set to true, the connection will not reconnect after a network failure. | Set to false and let StackExchange.Redis reconnect automatically. |
| ConnectRetry | The number of times to repeat connection attempts during initial connect. |  |
| ConnectTimeout | Timeout in ms for connect operations. |

In most cases the default values of the client are sufficient. You can fine tune the options based on your workload.

* Retries
  + For ConnectRetry and ConnectTimeout the general guidance is to fail fast and retry again. This is based on your workload and how much time on average it takes for your client to issue a Redis command and receive a response.
  + Let StackExchange.Redis automatically reconnect instead of checking connection status and reconnecting yourself. **Avoid using the ConnectionMultiplexer.IsConnected property**.
  + Snowballing - sometimes you may run into an issue where you are retrying and this snowballs and never recovers. In this case you should consider using an exponential backoff retry algorithm as described in [Retry general guidance](https://github.com/mspnp/azure-guidance/blob/master/Retry-General.md) published by the Microsoft Patterns & Practices group.
* Timeout values
  + Consider your workload and set the values accordingly. If you are storing large values, set the timeout to a higher value.
    - Set ABortOnConnectFail to false and let StackExchange.Redis reconnect for you.
* Use a single ConnectionMultiplexer instance for the application. You can use a LazyConnection to create a single instance that is returned by a Connection property, as shown in [Connect to the cache using the ConnectionMultiplexer class](https://msdn.microsoft.com/zh-cn/library/azure/dn690521.aspx#Connect).
* Set the ConnectionMultiplexer.ClientName property to an app instance unique name for diagnostic purposes.
* Use multiple ConnectionMultiplexer instances for custom workloads.
  + You can follow this model if you have varying load in your application. For example:
    - You can have one multiplexer for dealing with large keys.
    - You can have one multiplexer for dealing with small keys.
    - You can set different values for connection timeouts and retry logic for each ConnectionMultiplexer that you use.
    - Set the ClientName property on each multiplexer to help with diagnostics.
    - This will lead to more streamlined latency per ConnectionMultiplexer.

## What are some of the considerations when using common Redis commands?

* You should not run certain Redis commands which take a long time to complete without understanding the impact of these commands.
  + For example, do not run the [KEYS](http://redis.io/commands/keys) command in production as it could take a long time to return depending on the number of keys. Redis is a single-threaded server and it processes commands one at a time. If you have other commands issued after KEYS, they will not be processed until Redis processes the KEYS command.
* Key sizes - should I use small key/values or large key/values? In general it depends on the scenario. If your scenario requires larger keys then you can adjust the ConnectionTimeout and retry values and adjust your retry logic. From a Redis server perspective, smaller values are observed to have better performance.
  + This does not mean that you can’t store larger values in Redis; you must be aware of the following considerations. Latencies will be higher. If you have one set of data that is larger and one that is smaller, you can use multiple ConnectionMultiplexer instances, each configured with a different set of timeout and retry values, as described in the previous [What do the StackExchange.Redis configuration options do](#cache-configuration) section.

## When should I enable the non-SSL port for connecting to Redis?

Redis server does not support SSL out of the box, but Azure Redis Cache does. If you are connecting to Azure Redis Cache and your client supports SSL, like StackExchange.Redis, then you should use SSL.

Note that the non-SSL port is disabled by default for new Azure Redis Cache instances. If your client does not support SSL, then you must enable the non-SSL port by following the directions in the [Access ports](https://msdn.microsoft.com/zh-cn/library/azure/dn793612.aspx#AccessPorts) section of the [Configure a cache in Azure Redis Cache](https://msdn.microsoft.com/zh-cn/library/azure/dn793612.aspx) article.

Redis tools such as redis-cli do not work with the SSL port, but you can use a utility such as stunnel to securely connect the tools to the SSL port by following the directions in the [Announcing ASP.NET Session State Provider for Redis Preview Release](http://blogs.msdn.com/b/webdev/archive/2014/05/12/announcing-asp-net-session-state-provider-for-redis-preview-release.aspx) blog post.

For instructions on downloading the Redis tools, see the [How can I run Redis commands?](#cache-commands) section.

## How can I benchmark and test the performance of my cache?

* [Enable cache diagnostics](https://msdn.microsoft.com/zh-cn/library/azure/dn763945.aspx#EnableDiagnostics) so you can [monitor](https://msdn.microsoft.com/zh-cn/library/azure/dn763945.aspx) the health of your cache. You can view the metrics in the preview portal and you can also [download and review](https://github.com/rustd/RedisSamples/tree/master/CustomMonitoring) them using the tools of your choice.
* You can use redis-benchmark.exe to load test your Redis server.
  + Ensure that the load testing client and the Redis cache are in the same region.
* Use redis-cli.exe and monitor the cache using the INFO command.
  + If your load is causing high memory fragmentation then you should scale up to a larger cache size.
* For instructions on downloading the Redis tools, see the [How can I run Redis commands?](#cache-commands) section.

## How can I run Redis commands?

You can use any of the commands listed at [Redis commands](http://redis.io/commands#) except for the commands listed at [Redis commands not supported in Azure Redis Cache](/documentation/articles/cache-configure#redis-commands-not-supported-in-azure-redis-cache). To run Redis commands you have several options.

* If you have a Standard or Premium cache, you can run Redis commands using the [Redis Console](/documentation/articles/cache-configure#redis-console). This provides a secure way to run Redis commands in the preview portal.
* You can also use the Redis command line tools. To use them, perform the following steps.
  + Download the [Redis command line tools](https://github.com/MSOpenTech/redis/releases/download/win-2.8.19.1/redis-2.8.19.zip).
  + Connect to the cache using redis-cli.exe. Pass in the cache endpoint using the -h switch and the key using -a as shown in the following example.
    - redis-cli -h <your cache name>.redis.cache.chinacloudapi.cn -a <key>
  + Note that the Redis command line tools do not work with the SSL port, but you can use a utility such as stunnel to securely connect the tools to the SSL port by following the directions in the [Announcing ASP.NET Session State Provider for Redis Preview Release](http://blogs.msdn.com/b/webdev/archive/2014/05/12/announcing-asp-net-session-state-provider-for-redis-preview-release.aspx) blog post.

## What are some common cache patterns and considerations?

* Microsoft Patterns & Practices has the following guidance.
  + [Caching guidance](https://github.com/mspnp/azure-guidance/blob/master/Caching.md).
  + [Azure Cloud Application Design and Implementation Guidance](https://github.com/mspnp/azure-guidance)
* [Common cache patterns with Azure Redis Cache](/documentation/articles/cache-howto-common-cache-patterns)

## Why doesn’t Azure Redis Cache have an MSDN class library reference like some of the other Azure services?

Windows Azure Redis Cache is based on the popular open source Redis Cache, giving you access to a secure, dedicated Redis cache, managed by Microsoft. A variety of [Redis clients](http://redis.io/clients) are available for many programming languages. Each client has its own API that makes calls to the Redis cache instance using [Redis commands](http://redis.io/commands).

Because each client is different, there is not one centralized class reference on MSDN; instead each client maintains its own reference documentation. In addition to the reference documentation, there are several tutorials on Azure.com showing how to get started with Azure Redis Cache using different languages and cache clients on the [Redis Cache documentation](http://azure.microsoft.com/documentatgion/services/redis-cache/) page.